

Simple Termination of Context-Sensitive Rewriting

Salvador Lucas
Universidad Politécnica de Valencia
`slucas@dsic.upv.es`

joint work with

Bernhard Gramlich
Technische Universität Wien
`gramlich@logic.at`

Motivation: termination

- Proving termination of TRSs: find a **reduction** (i.e., a well-founded, monotone, and stable) **ordering** ' $>$ ' which is able to 'orientate' the rules of the TRS:

$$l > r \quad \text{for all rules } l \rightarrow r$$

- There is no general algorithm for that!

Motivation: termination

- Proving termination of TRSs **in practice**:
find a **simplification ordering** ' $>$ ' which is
able to 'orientate' the rules of the (often
transformed) TRS:

$$l > r \quad \text{for all rules} \quad l \rightarrow r$$

- Still, there is no general algorithm for that!
But most **automatizable orderings** are
simplification orderings!!

Motivation: programming languages

- **User-defined strategies** in rewriting-based programming languages (see Visser [WRS'01-ENTCSv57] for a survey):
 - ◆ ELAN,
 - ◆ Maude,
 - ◆ OBJ2,
 - ◆ OBJ3,
 - ◆ CafeOBJ, ...

Motivation: programming languages

Example: Consider the following OBJ program:

```
obj EXAMPLE is
  sorts Nat LNat .
  op 0      : -> Nat .
  op s      : Nat -> Nat .
  op nil    : -> LNat .
  op cons   : Nat LNat -> LNat [strat (1)] .
  op from   : Nat -> LNat .
  op sel    : Nat LNat -> Nat [strat (1 2 0)] .
  op first  : Nat LNat -> Nat [strat (1 2 0)] .
  vars N X  : Nat .
  var L     : LNat .
  eq sel(s(N),cons(X,L)) = sel(X,L) .
  eq sel(0,cons(X,L)) = X .
  eq first(0,L) = nil .
  eq first(s(N),cons(X,L)) = cons(X,first(N,L)) .
  eq from(N) = cons(N,from(s(N))) .
endo
```

Explicit 'local'
strategies

Missing arguments are
not evaluated!



Motivation: programming languages

*How to analyze the computational properties
of programs written in **such** languages?*

Motivation: programming languages

*Termination of rewriting and termination
of such programs do **not** coincide!!*

Motivation: programming languages

Example:

```
sel(s(0), from(0))      sel(s(0), cons(0, from(s(0))))
```

Terminating!?

```
sel(s(0), from(0))  
sel(s(0), cons(0, cons(s(0), from(s(s(0))))))  
sel(s(0), cons(0, s(0), s(s(0)), from(s(s(s(0)))))))  
...
```

Impossible (due to
replacement restrictions)!

Motivation: programming languages

Example: Consider the following OBJ program:

```
obj EXAMPLE is
  sorts Nat LNat .
```



```
vars N X      : Nat .
var L          : LNat .
eq sel(s(N),cons(X,L)) = sel(X,L) .
eq sel(0,cons(X,L)) = X .
eq first(0,L) = nil .
eq first(s(N),cons(X,L)) = cons(X,first(N,L)) .
eq from(N) = cons(N,from(s(N))) .
endo
```

Motivation: programming languages

Example: Consider the following OBJ program:

```
obj EXAMPLE is
  sorts Nat LNat .
```

Key: Syntactic replacement restrictions /
Context-sensitive rewriting

```
  vars N X      : Nat .
  var L         : LNat .
  eq sel(s(N),cons(X,L)) = sel(X,L) .
  eq sel(0,cons(X,L)) = X .
  eq first(0,L) = nil .
  eq first(s(N),cons(X,L)) = cons(X,first(N,L)) .
  eq from(N) = cons(N,from(s(N))) .
endo
```

Motivation: programming languages

■ Replacement maps:

sets (rather than lists) of positive integers associated to function symbols aimed at specifying which arguments are *reducible* in function calls

■ Context-Sensitive Rewriting (CSR [JFLP98]):

rewriting steps are limited by the replacement restrictions imposed by a replacement map

Motivation: programming languages

Example: Consider the following OBJ program:

```
obj EXAMPLE is
  sorts Nat LNat .
  op 0      : -> Nat
```

Sig

```
{0, s, nil, cons, sel,
  first, from}
```

**Replacement
map**

```
 $\mu(s) = \{1\}$   
 $\mu(\text{cons}) = \{1\}$   
 $\mu(\text{sel}) = \{1, 2\}$   
 $\mu(\text{from}) = \{1\}$   
 $\mu(\text{first}) = \{1, 2\}$ 
```

TRS

```
sel(s(N), cons(X, L)) -> sel(N, L)
sel(0, cons(X, L)) -> X
first(0, L) -> nil
first(s(N), cons(X, L)) -> cons(X, first(N, L))
from(N) -> cons(N, from(s(N)))
```

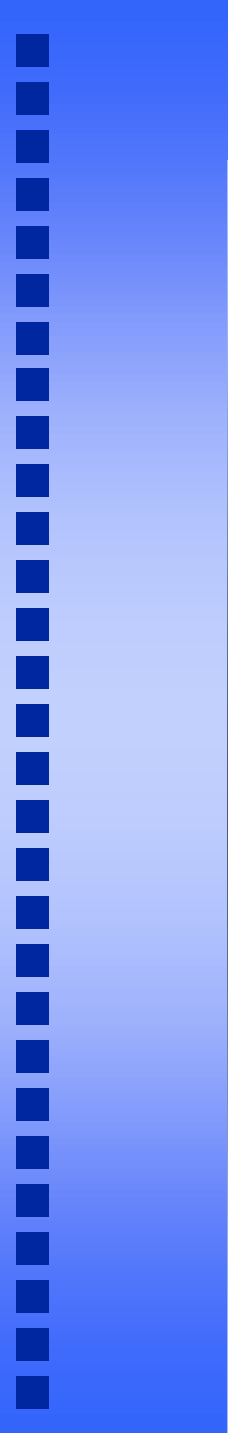
```
endo
```

Motivation: programming languages

*We develop a notion of simple termination of CSR which provides a unifying framework for **proving termination of CSR in practice***

Summary

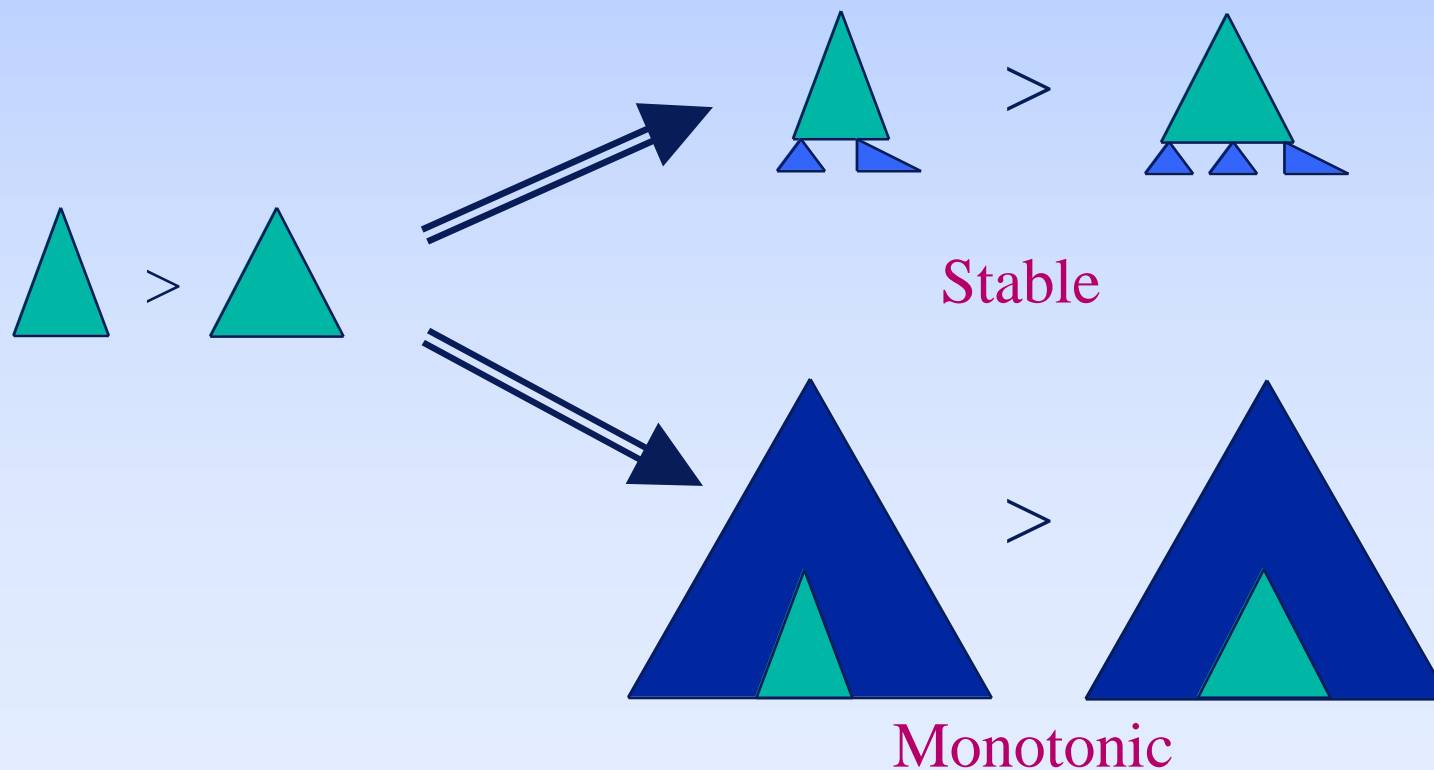
- Simple termination of rewriting
- Context-sensitive rewriting (CSR)
- Simple termination of CSR
- Proving simple termination of CSR
 - ◆ Transformation methods
 - ◆ Direct methods: CSRPO and polynomials
- Modularity of simple termination of CSR
- Conclusions and future work



Simple termination of rewriting

Simple termination of rewriting

- A reduction ordering is a well-founded, stable and monotonic ordering $>$



Simple termination of rewriting

- **Theorem:** A TRS R is **terminating** if and only if there is a **reduction ordering** $>$ such that

$$l > r$$

for all rules $l \rightarrow r$ in R

Simple termination of rewriting

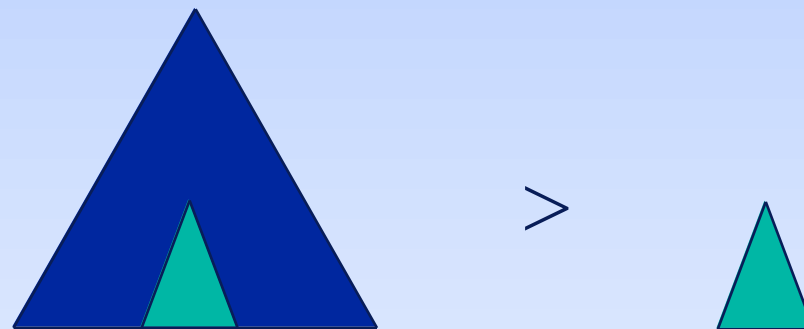
- **Definition** [MZ97]: A TRS R over a signature F is **simply terminating** if R extended with the rules

$$Emb(F) = \{f(x_1, \dots, x_i, \dots, x_k) \rightarrow x_i \mid f \text{ in } F \text{ and } 1 \leq i \leq k\}$$

is terminating

Simple termination of rewriting

- A **simplification ordering** is a **stable** and **monotonic** ordering $>$ with the **subterm property**:



Subterm

- The subterm property guarantees well-foundedness of $>$ (for finite signatures)

Simple termination of rewriting

- Examples of simplification orderings:
 - ◆ Knuth-Bendix ordering [KB70]
 - ◆ Recursive-path ordering(s) [Der82]
 - ◆ Polynomial orderings [Lan79]

Simple termination of rewriting

- **Theorem** [MZ97]: A TRS R over a finite signature F is **simply terminating** if and only if there is a **simplification ordering** $>$ such that

$$l > r$$

for all rules $l \rightarrow r$ in R



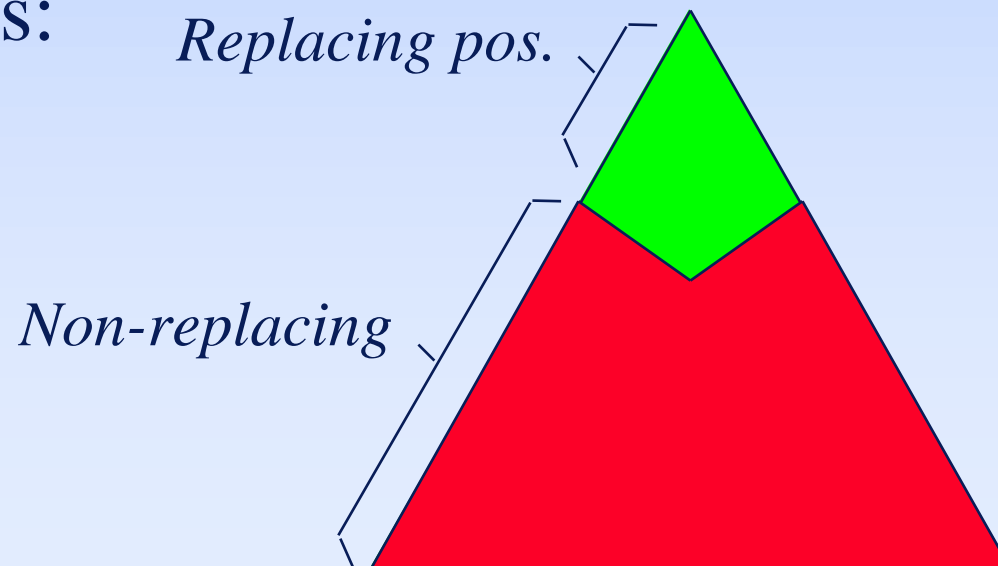
Context-sensitive rewriting

Context-sensitive rewriting

■ Replacement map:

A mapping μ which associates a subset $\mu(f)$ of $\{1, \dots, \text{ar}(f)\}$ to each function symbol f

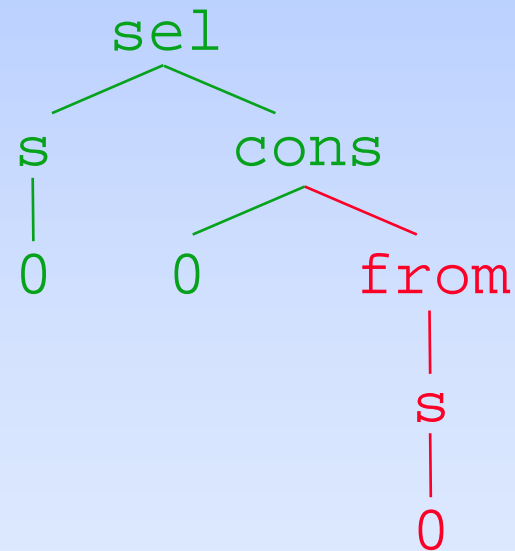
■ The set of positions of a term splits into two disjoint sets:



Context-sensitive rewriting

■ Example: $t = \text{sel}(s(0), \text{cons}(0, \text{from}(s(0))))$

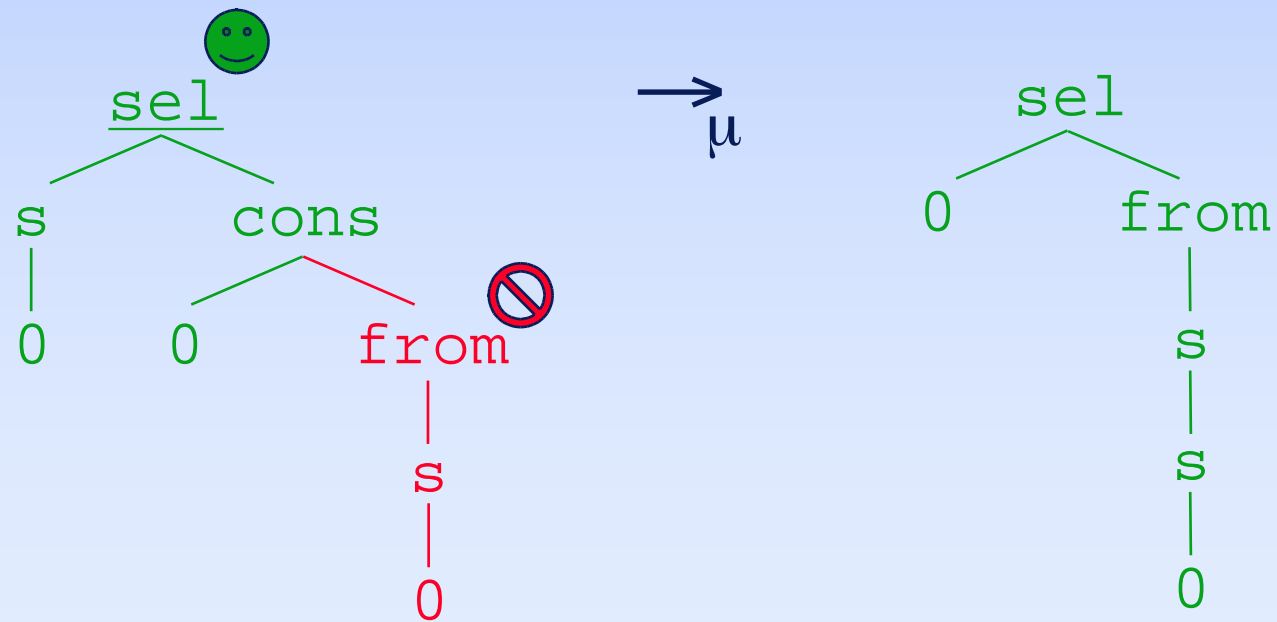
$\mu(0) =$
$\mu(s) = \{1\}$
$\mu(\text{nil}) =$
$\mu(\text{cons}) = \{1\}$
$\mu(\text{from}) = \{1\}$
$\mu(\text{sel}) = \{1, 2\}$



Context-sensitive rewriting

■ Context-sensitive rewriting:

Given a replacement map μ , *only* rewriting steps at replacing positions are allowed (written $t \rightarrow_{\mu} s$)

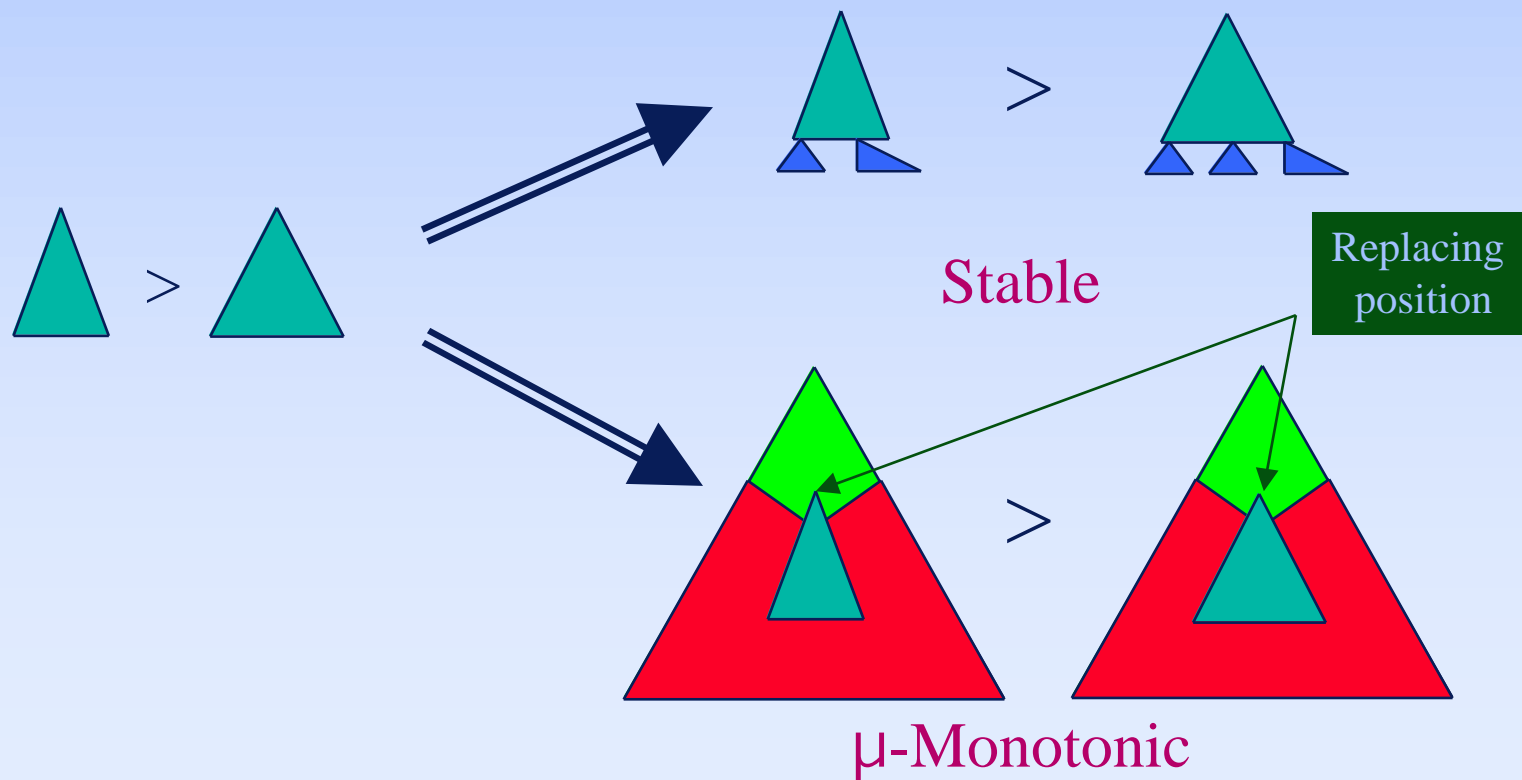




Simple termination of CSR

Simple termination of CSR

- A μ -reduction ordering is a well-founded, stable and μ -monotonic ordering > [Zan97]:



Simple termination of CSR

- **Theorem** [Zan97]: A TRS R is μ -terminating if and only if there is a μ -reduction ordering $>$ such that

$$l > r \text{ for all rule } l \rightarrow r \text{ in } R$$

Simple termination of CSR

- **Definition:** A TRS R over a signature F is **simply μ -terminating** if R extended with the rules

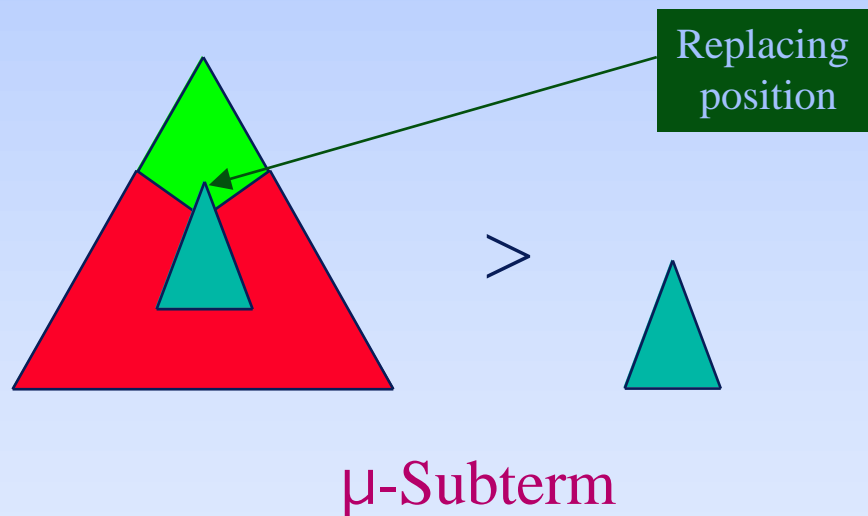
$$Emb^{\mu}(F) = \{f(x_1, \dots, x_i, \dots, x_k) \rightarrow x_i \mid f \text{ in } F \text{ and } i \text{ in } \mu(f)\}$$

is **μ -terminating**



Simple termination of CSR

- A μ -simplification ordering is a *stable* and μ -monotonic ordering $>$ with the μ -subterm property



- The μ -subterm property does **not** guarantee well-foundedness of $>$ (!!)

Simple termination of CSR

- **Example:** Consider the constant a and the monadic symbol c together with $\mu(c) = \dots$. Then, the ordering $>$ given by

$$c^n(a) > c^{n+1}(a) \quad \text{for } n=0, 1, \dots$$

is stable, μ -monotonic and has the μ -subterm property, but it is **not well founded**:

$$a > c(a) > c(c(a)) > \dots$$

Simple termination of CSR

- **Theorem:** A TRS R is simply μ -terminating if and only if there is a **well-founded μ -simplification ordering** $>$ such that
$$l > r \text{ for all rule } l \rightarrow r \text{ in } R$$



Proving simple termination of CSR



Transformation methods

Proving simple termination of CSR

- **Transformations:** The μ -termination of a TRS R is demonstrated by proving termination of a TRS R^μ_Θ for a given transformation Θ :
 - ◆ Lucas [ICALP96]
 - ◆ Zantema [RTA97]
 - ◆ Ferreira and Ribeiro [RTA99]
 - ◆ Giesl and Middeldorp [RTA99] (gave a **complete** transformation)

Proving simple termination of CSR

- Lucas' transformation [ICALP96]: **remove** all non-replacing subterms from the rules of the TRS R : For instance:

$$R \left\{ \begin{array}{l} \text{first}(0, x) \rightarrow \text{nil} \\ \text{first}(s(x), \text{cons}(y, z)) \rightarrow \text{cons}(y, \text{first}(x, z)) \\ \text{from}(x) \rightarrow \text{cons}(x, \text{from}(s(x))) \end{array} \right.$$

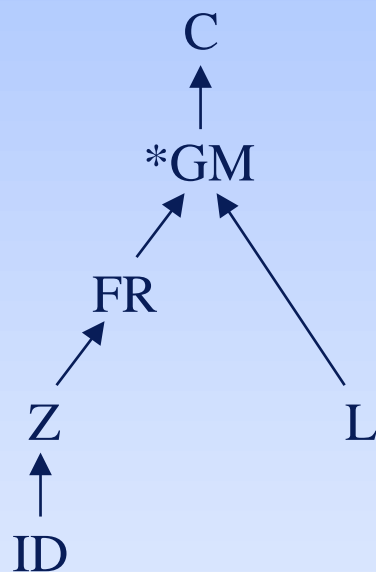
$$\mu(\text{cons}) = \mu(\text{from}) = \mu(s) = \{1\}, \mu(\text{first}) = \{1, 2\}$$

$$R_L^\mu \left\{ \begin{array}{l} \text{first}(0, x) \rightarrow \text{nil} \\ \text{first}(s(x), \text{cons}(y)) \rightarrow \text{cons}(y) \\ \text{from}(x) \rightarrow \text{cons}(x) \end{array} \right.$$

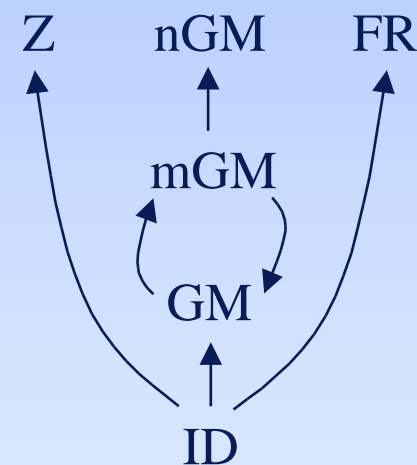
Note that R_L^μ is terminating (e.g., use *RPO*)

Proving simple termination of CSR

■ Comparison:



Termination [GM99]



Simple term. [RTA02]

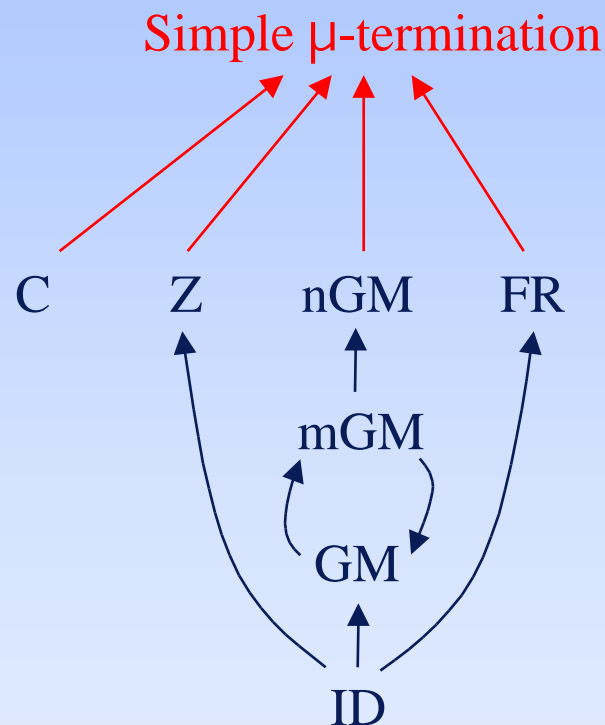
The transformations are available for use within MU-TERM 1.0:

<http://www.dsic.upv.es/users/elp/slucas/muterm>

Proving simple termination of CSR

What about proving
simple μ -termination?

Proving simple termination of CSR



Proving simple μ -termination using transformations



Direct methods

Proving simple termination of CSR

- **Direct methods for proving μ -termination:**
 - ◆ CSRPO (Borralleras, Lucas & Rubio [CADE02])
- RPO is a well-known simplification ordering
- CSRPO **generalizes RPO** to cope with context-sensitive replacement restrictions
- **Theorem:** CSRPO is a well-founded μ -simplification ordering

Proving simple termination of CSR

- **Polynomials**: The μ -termination of the following TRS R [GM99]:

$$\boxed{\mu(f)=\{3\}} \quad \left\{ \begin{array}{l} f(a, b, x) \rightarrow f(x, x, x) \\ c \rightarrow a \\ c \rightarrow b \end{array} \right.$$

cannot be proved by using transformations (or CSRPO)!

- We consider polynomial interpretations using **integer** coefficients and returning nonnegative values (and requiring **μ -monotonicity** only)

Proving simple termination of CSR

- **Definition:** Polynomial μ -orderings are those induced by polynomial μ -interpretations
- **Theorem:** polynomial μ -orderings are **well-founded μ -simplification orderings**

Proving simple termination of CSR

- In the previous example, we let:

$$\begin{aligned} f(x,y,z) &= 1 + (x - y)^2 + z & a &= 2 \\ c &= 3 & b &= 1 \end{aligned}$$

- Then, we have

$$[f(a, b, x)] = 2 + x > 1 + x = [f(x, x, x)]$$

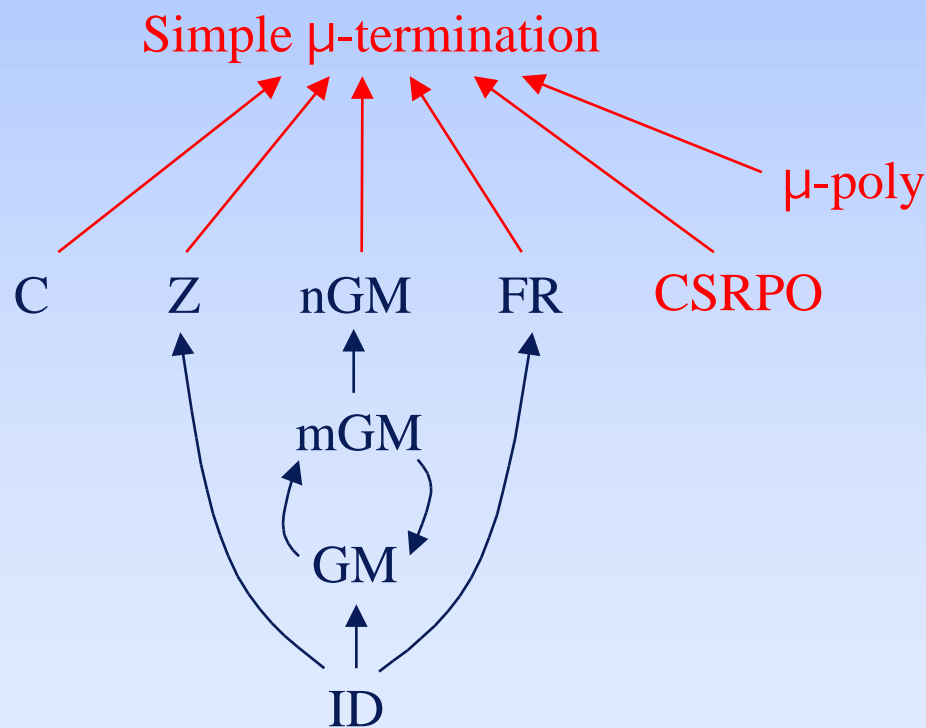
$$[c] = 3 > 2 = [a]$$

$$[c] = 3 > 1 = [b]$$

Therefore, ***R*** is simply μ -terminating!

Proving simple termination of CSR

■ Comparison:



Proving simple μ -termination



Modularity of simple termination of CSR

Modularity of simple μ -termination

■ **Definition:** A **CSRS** is a pair (R, μ) , where R is a TRS and μ is a replacement map

■ **Theorem:** Simple termination of CSR is **modular** for disjoint unions of CSRSs

A (necessary?) condition: *there must be symbols f in the signature of each module which have at least **two** replacing arguments*

Modularity of simple μ -termination

- **Theorem:** Simple termination of CSR is **modular** for constructor-sharing unions of CSRSs where the **shared constructors** are **fully replacing**

*Again: there must be symbols f in the signature of each module which have at least **two replacing arguments***

- Without fully replacing shared constructors the theorem does **not** hold!

Modularity of simple μ -termination

■ Example: Consider the CSRSs

$$R \left\{ \begin{array}{l} \text{zeros} \rightarrow 0:\text{zeros} \\ \mu(\cdot)=\{1\} \end{array} \right.$$

simply μ -terminating

$$R' \left\{ \begin{array}{l} \text{length}([\]) \rightarrow 0 \\ \text{length}(x:y) \rightarrow s(\text{length}(y)) \\ \mu(\cdot)=\mu(\text{length})=\mu(s)=\{1\} \end{array} \right.$$

simply μ -terminating

$$R + R' \left\{ \begin{array}{l} \text{zeros} \rightarrow 0:\text{zeros} \\ \text{length}([\]) \rightarrow 0 \\ \text{length}(x:y) \rightarrow s(\text{length}(y)) \\ \mu(\cdot)=\mu(\text{length})=\mu(s)=\{1\} \end{array} \right.$$

non- μ -terminating!

$\text{length}(\underline{\text{zeros}}) \rightarrow_{\mu} \underline{\text{length}(0:\text{zeros})} \rightarrow_{\mu} s(\text{length}(\underline{\text{zeros}})) \rightarrow_{\mu} \dots$



Conclusions and future work

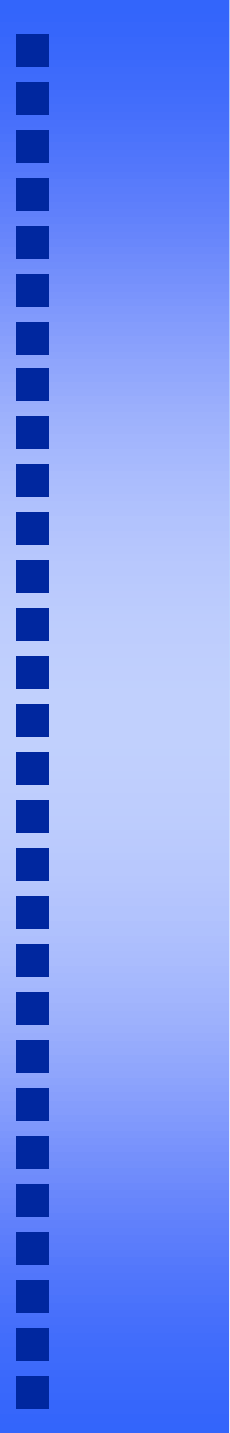
Conclusions

- We have introduced:
 - ◆ the notions of simplification ordering and simple termination for CSR
 - ◆ polynomial orderings for CSR
 - ◆ preliminary results regarding modularity of simple termination of CSR
- Simple termination of CSR provides a **unifying** framework for proving termination of CSR in practice



Future work

- Well-foundedness of μ -simplification orderings (Kruskal)
- CSKBO (Borralleras [PhD02])
- Modularity in more general combinations of CSRSs (e.g., composable, hierarchical)



Simple Termination of Context-Sensitive Rewriting

Salvador Lucas
Universidad Politécnica de Valencia
`slucas@dsic.upv.es`

joint work with

Bernhard Gramlich
Technische Universität Wien
`gramlich@logic.at`